Unit-01 Introduction to Data Structure

Topic to be covered

- Array:
  - Representation of arrays
    - One dimensional array
    - Two dimensional array
  - Applications of arrays
    - Symbol Manipulation (matrix representation of polynomial equation)
    - Sparse matrix
  - Sparse matrix and its representation

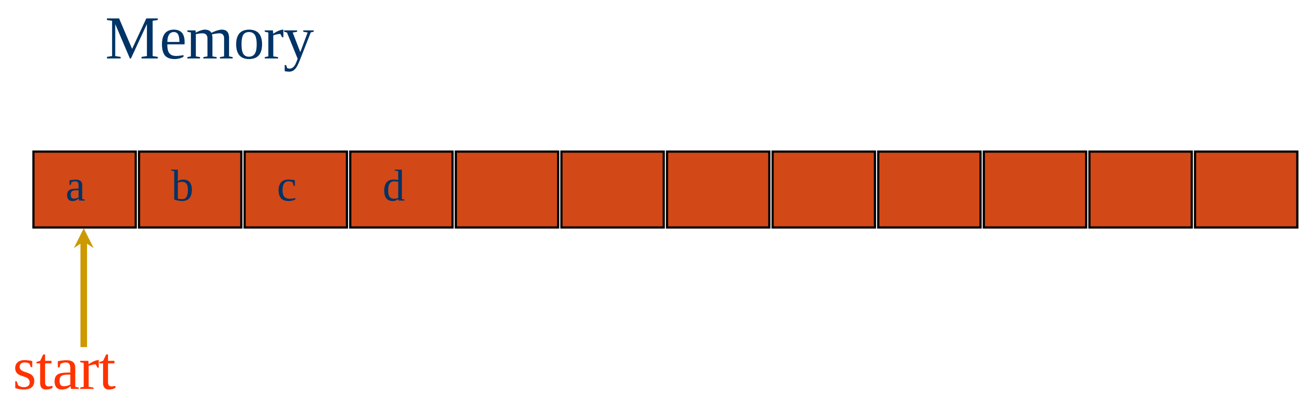
1- D Array

- Simplest data structure that makes use of computer address to locate its elements is the 1-dimensional array or vector

- Number of memory locations is sequentially allocated to the vector

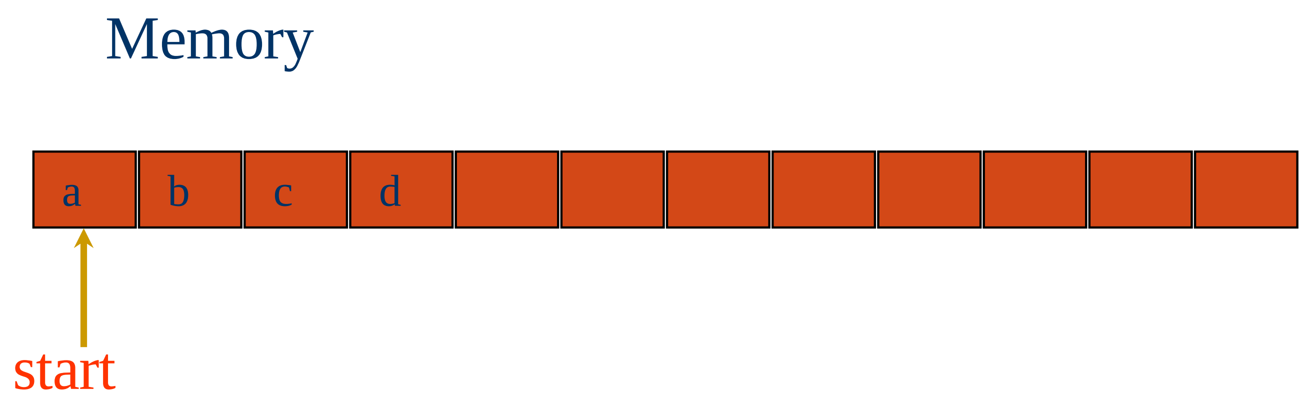
- A vector size is fixed and therefore requires a fixed number of memory locations

1-D Array Representation



- 1-dimensional array x = [a, b, c, d]
- map into contiguous memory locations
- location(x[i]) = start + i

Space Overhead



space overhead = 4 bytes for start

1-D Array Size

- A array will store five integer values, its name is num.
- it can be visualized as show.

index	data
num[0]	10
num[1]	20
num[2]	30
num[3]	40
num[4]	50

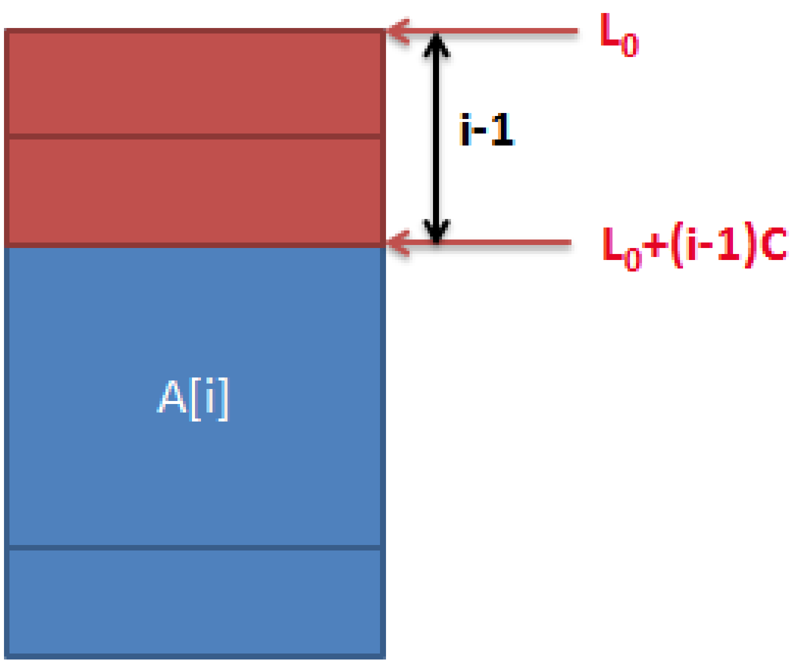
size of array=(upper bound – lower bound) +1

=4+0+1

=5

1-D Array Location

- Array A with subscript lower bound of "one" is represented as below.



- L_0 is the address of the first word allocated to the first element of vector A
- C words is size of each element or node
- The address of element Ai is

  $Loc(Ai)=L_0+(C*(i-1))$

- Let's consider the more general case of a vector A with lower bound for it's subscript is given by some variable b.
- The address of element Ai is

  $Loc\ (Ai) = L_0 + (C*(i-b))$

2-D Array

- 2-D arrays are also called table or matrix
- 2-D arrays have two subscripts- row & column

2-D Arrays

The elements of a 2-dimensional array a declared as:

```
int a[3][4];
```

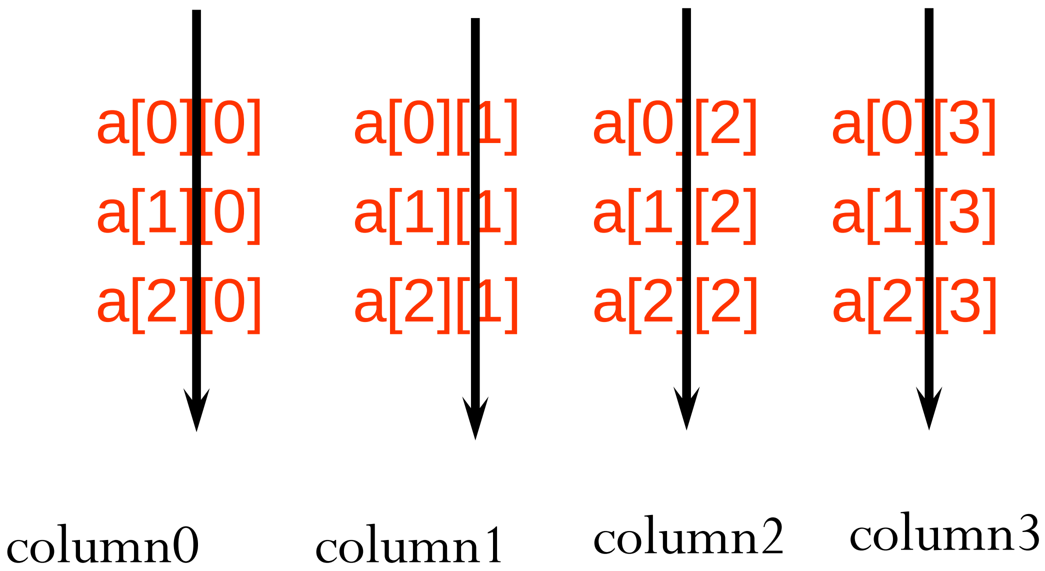
may be shown as a table

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

Rows of a 2-D Array

a[0][0]	a[0][1]	a[0][2]	a[0][3]	row 0
a[1][0]	a[1][1]	a[1][2]	a[1][3]	row 1
a[2][0]	a[2][1]	a[2][2]	a[2][3]	row 2

Columns of a 2-D Array



2-D Array

- **Initialization of Two dimensional arrays:**

```
int arr[2][3]={
        {10,20,30},
        {40,50,60}
    };
```

- Bellow show the data in 2-D grid.

	0	1	2
0	10	20	30
1	40	50	60

Implementation of 2-D Array in memory

- A two-dimensional array can be implemented in a programming language in two ways:
  - Row-major implementation/ Row-major order matrix
  - Column-major implementation/ Column-major order matrix

Row-major order matrix

- 2-D array in which elements are stored row by row

For example an array [3][3] is stored in the memory as show bellow:

A00	A01	A02	A10	A11	A12	A20	A21	A22

Row 1 Row 2 Row 3

2-D array row-major order matrix

The storage can be clearly understood by arranging array as matrix:

$$\begin{array}{cc} & \begin{array}{cccc} b2 & & & u2 \end{array} \\ \begin{array}{c} b1 \\ \\ u1 \end{array} & \left(\begin{array}{cccc} [1,1] & [1,2] & [1,3] & [1,m] \\ [2,1] & [2,2] & [2,3] & [2,m] \\ [n,1] & [n,2] & [n,3] & [n,m] \end{array}\right)_{nxm} \end{array}$$

n = no of rows, **m** = no of columns

b1 = lower bound subscript of row
u1 = upper bound subscript of row
n = u1 – b1 + 1

b2 = lower bound subscript of column
u2 = upper bound subscript of column
m = u2 – b2 + 1

The address element A [i , j] is given by

$$Loc (A [i , j]) = L_0 + (i-b1)*(u2-b2+1) + (j – b2)$$

With Storage space

- storage space of each element is represented by W

Then formula will be-

$$Loc(A[i, j]) = L_0 + W(m*(i-b_1)+(j-b_2))$$

Example-

A 2-D array defined as a[4:7,-1:3] requires 2 bytes of storage space for each element. If the array is stored in row-major form , then calculate the address of element at location a[6,2] given base address is 100.

- **Sol:** $L_0=100$, $b_1=4$, $b_2=-1$, $u_1=7$, $u_2=3$, $W=2$(int size)
- $i=6$, $j=2$ and m(no. of col)= $u_2- b_2+1=3-(-1)+1=5$
- Address of a[i][j]= $L_0+W(m(i-b_1)+(j-b_2))$
- Address of a[6][2]=100+2(5(6-4) + (2-(-1))

  =100+(2(5*2+3)

  =100+26

  =126

2-D array column-major order matrix

- 2-D array in which elements are stored column by column

- For example an array [3][3] is stored in the memory as show bellow:

A00	A10	A20	A01	A11	A21	A02	A12	A22

Col 1 Col 2 Col 3

a=

A00	A01	A02
A10	A11	A12
A20	A21	A22

Col 1 Col 2 Col 3

2-D array column-major order matrix

$$\begin{array}{c c} & \begin{array}{cccc} \text{Col-1} & \text{Col-2} & \text{Col-3} & \text{Col-4} \end{array} \\ \begin{array}{c} \text{Row 1} \\ \text{Row 2} \end{array} & \begin{bmatrix} [1,1] & [1,2] & [1,3] & [1,4] \\ [2,1] & [2,2] & [2,3] & [2,4] \end{bmatrix} \end{array}$$

The address of element A [i , j] can be obtained by expression

Loc (A [i , j]) = L_0 + (j-1)*2 + (i-1)

Loc (A [2, 3]) = L_0 + (3-1)*2 + (2-1) = $L_0 + 5$

2-D array column-major order matrix

In general for two dimensional array consisting of **n rows** and **m columns** the address element A [i , j] is given by

$$Loc(A[i, j]) = L_0 + W(n*(j-b_2) + (i-b_1))$$

W=size of each element array element
n=the number of row (u_1-b_1+1)
b_1 the lower bound of row
u_1 upper bound of row
b_2 is lower bound of column
u_2 upper bound of column

2-D Arrays:
Column-major Implementation

Example-

A two-dimensional array defined as a[-20:20,10:35] requires one bytes of storage space for each element. If the array is stored in column-major form , then calculate the address of element at location a[0,30] given base address is 500.

Solution-

$B=500$, $b_1= -20$, $b_2=10$, $u_1=20$, $u_2=35$

$W=1$ byte

$i=0, j=30$ and n (no. of rows)$= u_1 - b_1+1$

$=20-(-20)+1=41$

Address of $a[i][j]= B+W(n(j-b_2)+(i-b_1))$

Address of $a[0][30]=500+1(41(30-10)+(0-(-20))$

$=500+1(41*20+20)$

$=500+1(820+20)$

$=500+840$

$=1340$

Q-1

Let A be a two-dimensional array declared as follows:

A: array [1 …. 10] [1 …… 15] of integer;

Assuming that each integer takes one memory location, the array is stored in **row-major** order and the first element of the array is stored at location 100, what is the address of the element A[7][12]?

$$
\begin{aligned}
\text{Loc } A[i][j] &= 100 + [\,(i - 1)\,(15 - 1 + 1) + (j - 1)\,] \times 1 \\
&= 100 + (i - 1)\,15 + j - 1 \\
&= 15i + j + 84 \\
&= 15*7 + 12 + 84 \\
&= \mathbf{201}
\end{aligned}
$$

Multi-Dimensional Array

- The general form of a multi-dimensional array is:

  type array_name[s1][s2][s3]…..[sm];

- Suppose 3-D array can be group of an array of arrays.

- For example : int a[3][3][2];

- Here 3-D array which is collection of three 2-D arrays each contain 3 rows and 2 column.

Multi-Two-Dimensional Array

- For example:

```
int a[3][3][2]={ { {1,2},{3,4},{5,6}},
                 { {1,2},{3,4},{5,6}},
                 { {1,2},{3,4},{5,6}}
               };
```

- Memory representation of above 3-D array is bellow:

1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6

0th 2-D array   1st 2-D array   2nd 2-D array

Different Implementations

- Row Major-

$$Loc(A[k, i, j]) = L_0 + [n*m(k-K\ ) + m(i-L1) + (j-L2)]*W$$

- Column Major-

$$Loc(A[k, i, j]) = L_0 + [n*m(k-K) + n(j-L2) + (i-L1)]*W$$

N=no of rows, m=no of column, K= Lowerbound,
L1= lower bound of row, L2= lower bound of column

Example

Let A be a 3-dimensional array declared as follows:

A: array [1 …. 8] [1 …… 5] [1……7] of integer;

Given $L_0 = 900$

Find location of A[5,3,6] element using row major & column major?

Applications of Array

1. Symbol Manipulation (matrix representation of polynomial equation)
2. Sparse Matrix

Matrix representation of Polynomial Equation

- We can use array for different kind of operations in polynomial equation such as addition, subtraction etc.

- Array can be used to represent polynomial equation

Example

	Y	Y^2	Y^3	Y^4
X	XY	XY^2	XY^3	XY^4
X^2	X^3Y	X^2Y^2	X^2Y^3	X^2Y^4
X^3	X^3Y	X^3Y^2	X^3Y^3	X^3Y^4
X^4	X^4Y	X^4Y^2	X^4Y^3	X^4Y^4

$2X^2 + 5XY + Y^2$

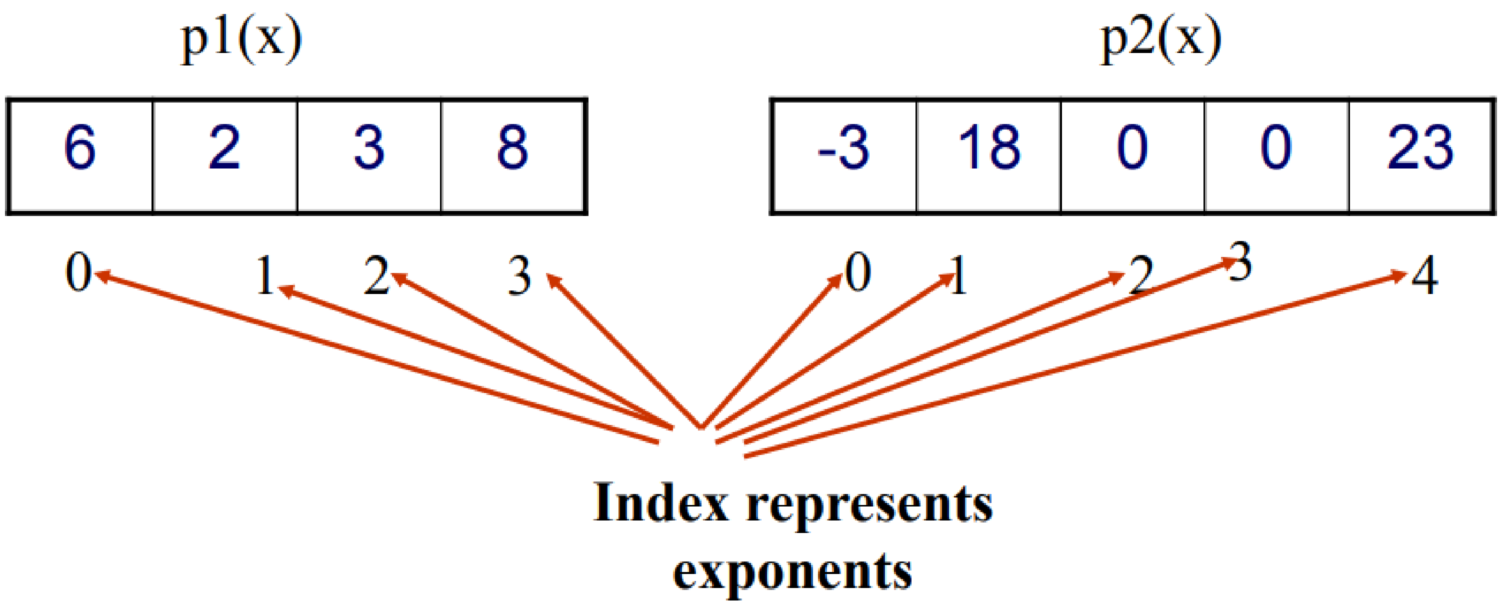
		Y	Y^2	Y^3	Y^4
	0	0	1	0	0
X	0	5	0	0	0
X^2	2	0	0	0	0
X^3	0	0	0	0	0
X^4	0	0	0	0	0

$X^2 + 3XY + Y^2 + Y - X$

		Y	Y^2	Y^3	Y^4
	0	1	1	0	0
X	-1	3	0	0	0
X^2	1	0	0	0	0
X^3	0	0	0	0	0
X^4	0	0	0	0	0

Polynomial ADT

- Array Implementation:
- $p1(x) = 8x^3 + 3x^2 + 2x + 6$
- $p2(x) = 23x^4 + 18x - 3$



Why arrays aren't good to represent polynomials?

- $p3(x) = 16x^{21} - 3x^{5} + 2x + 6$

6	2	0	0	-3	0	…………	0	16

<u>WASTE OF SPACE!</u>

Exercise-

1. Write an algorithm to add two polynomials using linked list.

2. Write an algorithm to add two polynomials using Array.

Sparse Matrix

- An mXn matrix is said to be ***sparse*** if "many" of its elements are zero.
- A matrix is not sparse is called a ***dense matrix***.
- We can device a simple representation scheme whose space requirement equals the size of the non-zero elements.

$$
\begin{array}{r|cccccccc|}
 & \text{Column - 1} & \text{Column - 2} & \text{Column - 3} & \text{Column - 4} & \text{Column - 5} & \text{Column - 6} & \text{Column - 7} & \text{Column - 8} \\
\hline
\text{Row - 1} & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\
\text{Row - 2} & 0 & 6 & 0 & 0 & 7 & 0 & 0 & 3 \\
\text{Row - 3} & 0 & 0 & 0 & 9 & 0 & 8 & 0 & 0 \\
\text{Row - 4} & 0 & 4 & 5 & 0 & 0 & 0 & 0 & 0 \\
\end{array}_{4x8}
$$

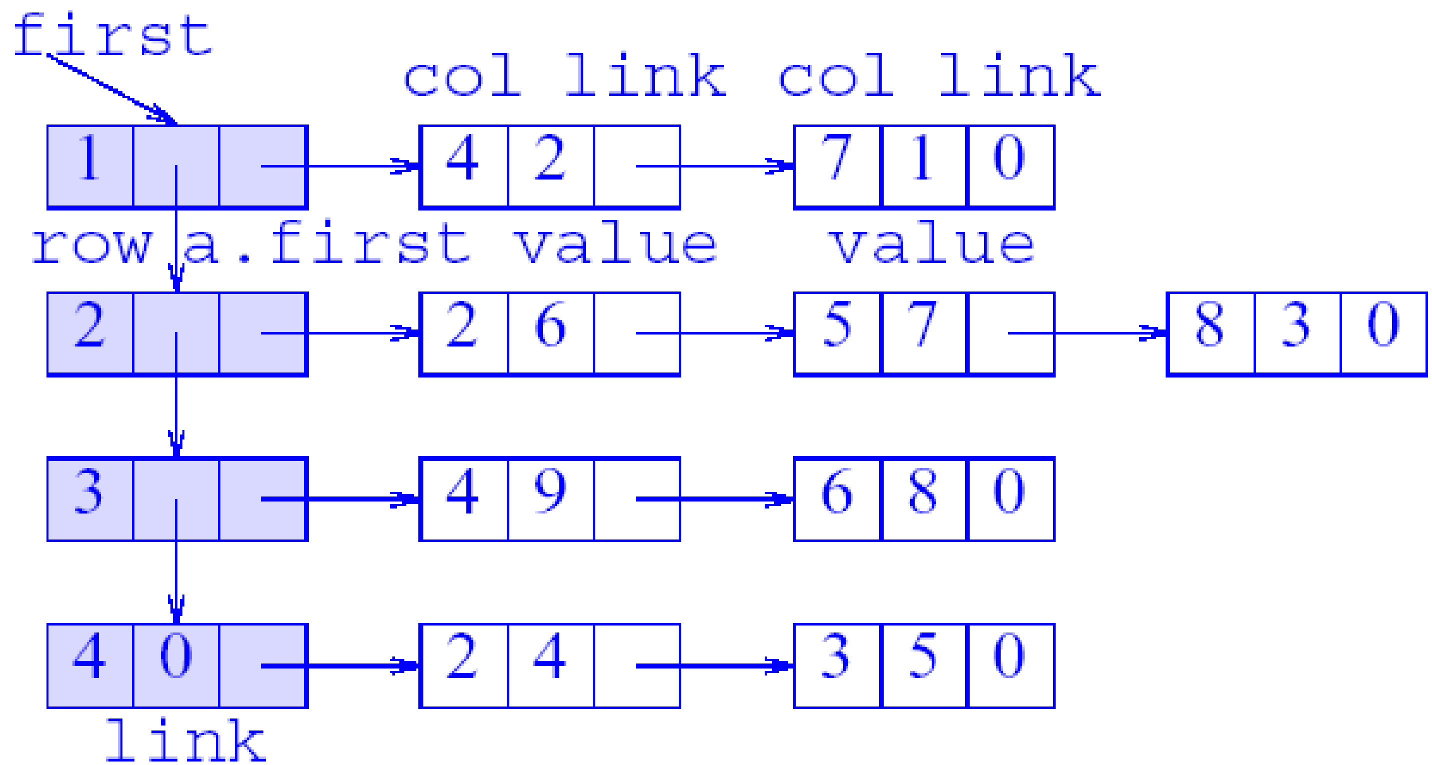
Terms	0	1	2	3	4	5	6	7	8
Row	1	1	2	2	2	3	3	4	4
Column	4	7	2	5	8	4	6	2	3
Value	2	1	6	7	3	9	8	4	5

Linear Representation of given matrix

Any shortcoming of the 1-D array of a sparse matrix representation?

We need to know the number of nonzero terms in each of the sparse matrices when the array is created.

Linked Representation of Sparse Matrix



Sparse Matrix

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & 0 & 6 & 0 & 9 & 0 & 0 \\ 2 & 0 & 0 & 7 & 8 & 0 & 4 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 5 \end{pmatrix}_{6x7} \end{array}$$

Memory Space required to store 6x7 matrix

42 x 2 = 84 bytes

Memory Space required to store Linear Representation

30 x 2 = 60 bytes

Linear representation of Matrix

Row	Column	A
1	3	6
1	5	9
2	1	2
2	4	7
2	5	8
2	7	4
3	1	10
4	3	12
6	4	3
6	7	5

Space Saved = 84 – 60 = 24 bytes

Sparse Matrix

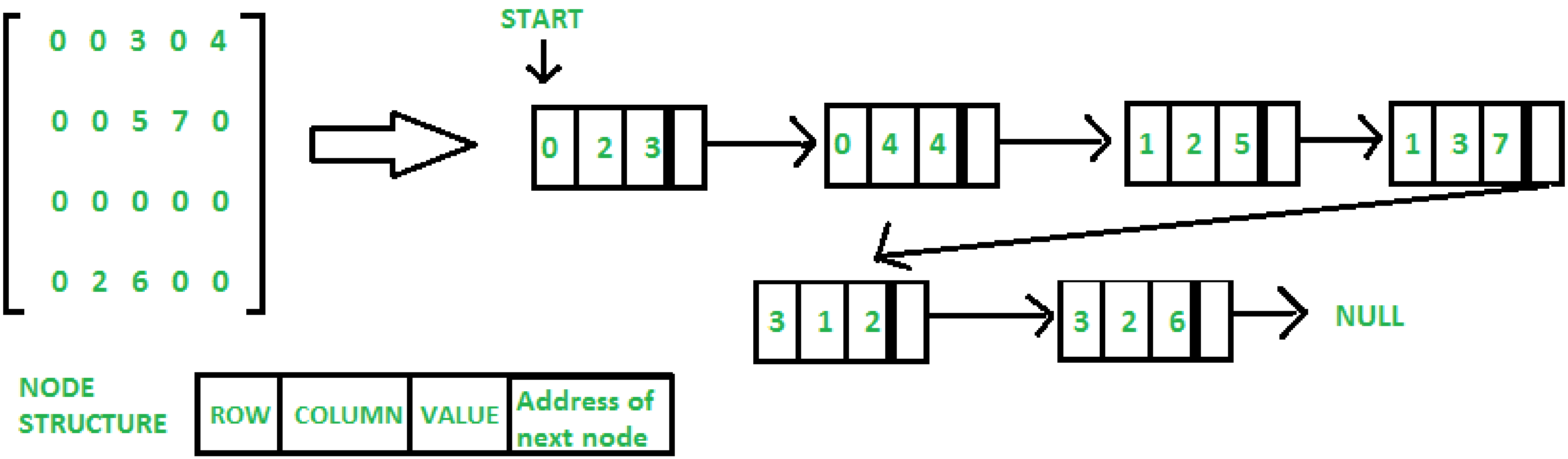
Linear representation of Matrix

Row	Column	A
1	3	6
1	5	9
2	1	2
2	4	7
2	5	8
2	7	4
3	1	10
4	3	12
6	4	3
6	7	5

	Column	A
1	3	6
2	5	9
3	1	2
4	4	7
5	5	8
6	7	4
7	1	10
8	3	12
9	4	3
10	7	5

Memory Space required to store Liner Representation = 26 x 2 = 42 bytes

Linked List representation of sparse matrix



Case study-

- A super market conducting a study of the mix items purchased by its customers.
- For this study data are gathered for the purchase made by 1000 customers.
- These data are organized into a matrix, purchases with purchases(i,j) being the quantity of item i purchased by customer j.
- Suppose that the super market has an inventory of 10,000 different items.
- The purchase matrix is therefore a 10,000 x 1,000 matrix
- If the average customer buys 20 different items only about 20,000 of 1,00,000,000 matrix entries are nonzero

THANK YOU